Algorithms and Applications for Predictive Search

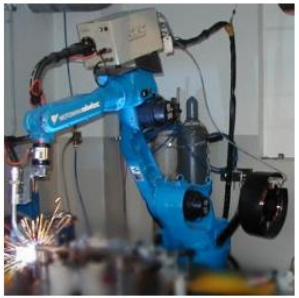
Rudolf Kruse
Faculty of Computer Science
Otto von Guericke University Magdeburg

Alexander Dockhorn
School of Electrical and Computer Engineering
Queen Mary University London

Motivation

- Computational Intelligence refers to the ability of a computer to **learn** a specific task **from data** or **experimental observation**.
  - steering of self-driving cars and parking aids
  - automating a production pipeline

[1]



[2]

OTTO VON GUERICKE UNIVERSITÄT MAGDEBURG

Queen Mary University of London

Motivation

Problems with real environments

- Real tasks are hard to setup
- Failure of the algorithm has considerable cost
- It becomes hard to study the algorithms underlying characteristics

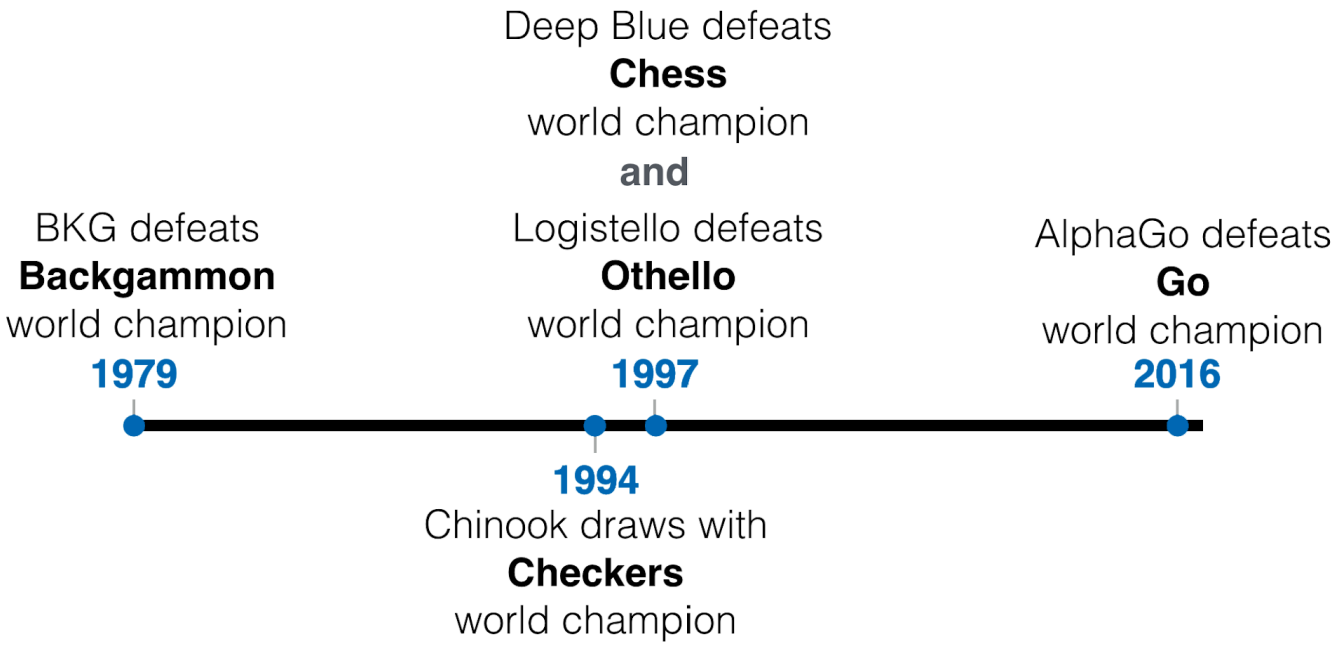
Games can be simulations of real world tasks

- Quantifiable goals, controllable difficulty, and large data sets
- Digital games are fully accessible to computers

OTTO VON GUERICKE UNIVERSITÄT MAGDEBURG

Queen Mary University of London

A Short History of Computational Intelligence in Games



OTTO VON GUERICKE UNIVERSITÄT MAGDEBURG

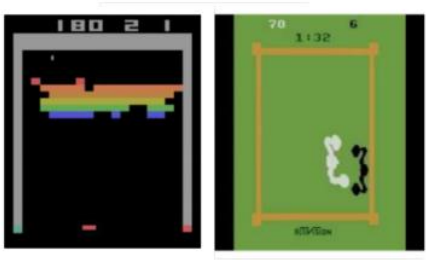
Queen Mary University of London

Modern Reinforcement Learning Examples

ATARI

StarCraft II

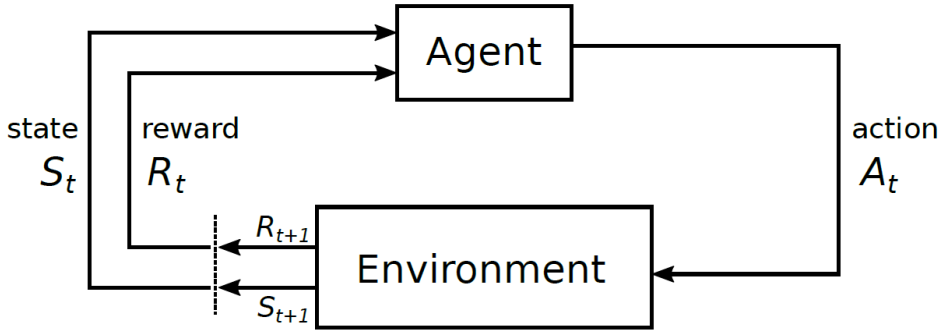
DOTA 2



OTTO VON GUERICKE UNIVERSITÄT MAGDEBURG

Queen Mary University of London

Agent–Environment Interface



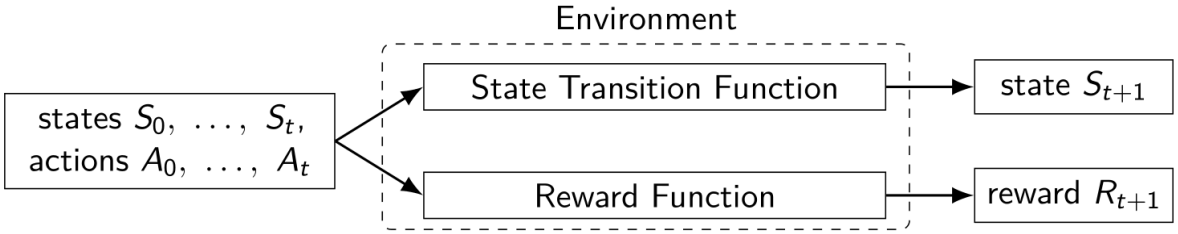
A general framework for studying games which consists of the elements

- **Agent:** the learner and decision-maker
- **Environment:** anything the agent interacts with, e.g. a game
- **Actions:** Agent and environment interact continuously interact with each other.
- **Reward:** numerical values provided that the agent tries to maximize over time.

OTTO VON GUERICKE UNIVERSITÄT MAGDEBURG

Queen Mary University of London

Components of a Game



State $S_t \in \mathcal{S}$ can be perceived through multiple sensors $(S_t^{(1)}, S_t^{(2)}, \ldots, S_t^{(n)})$.

- a state may not be fully observable (partial information game)

The whole environment can be modelled as a probability distribution of possible outcomes:

$$P(R_{t+1}, S_{t+1} \mid S_0, A_0, S_1, A_1, \ldots, S_t, A_t)$$

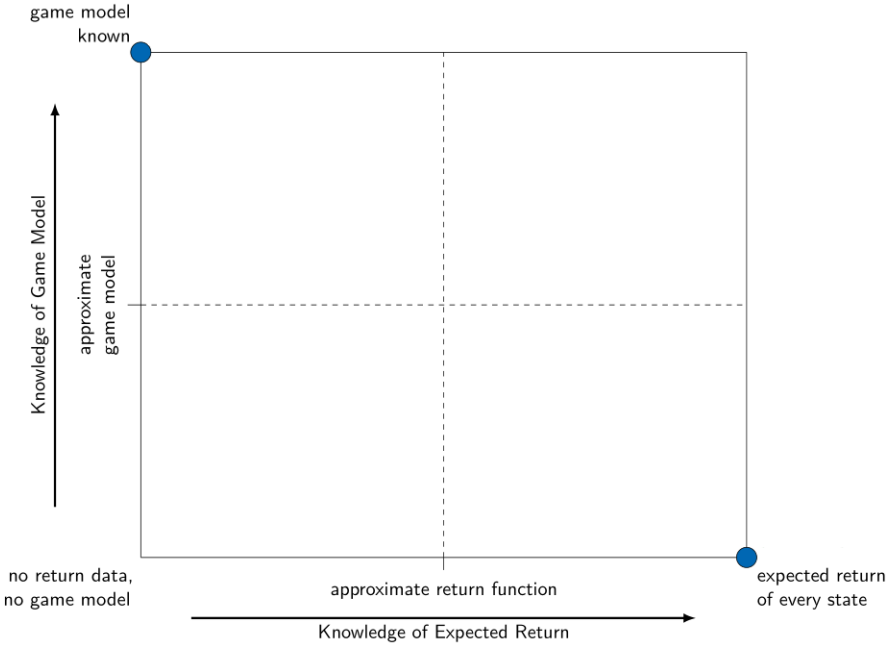
- But we can also model both components separately

Problem Classification

Two popular learning approaches:

1) Learn which actions are good
2) Learn to anticipate the future

Both allow the definition of learning algorithms as well as approximate solutions and hybrid algorithms.



OTTO VON GUERICKE UNIVERSITÄT MAGDEBURG

Queen Mary University of London

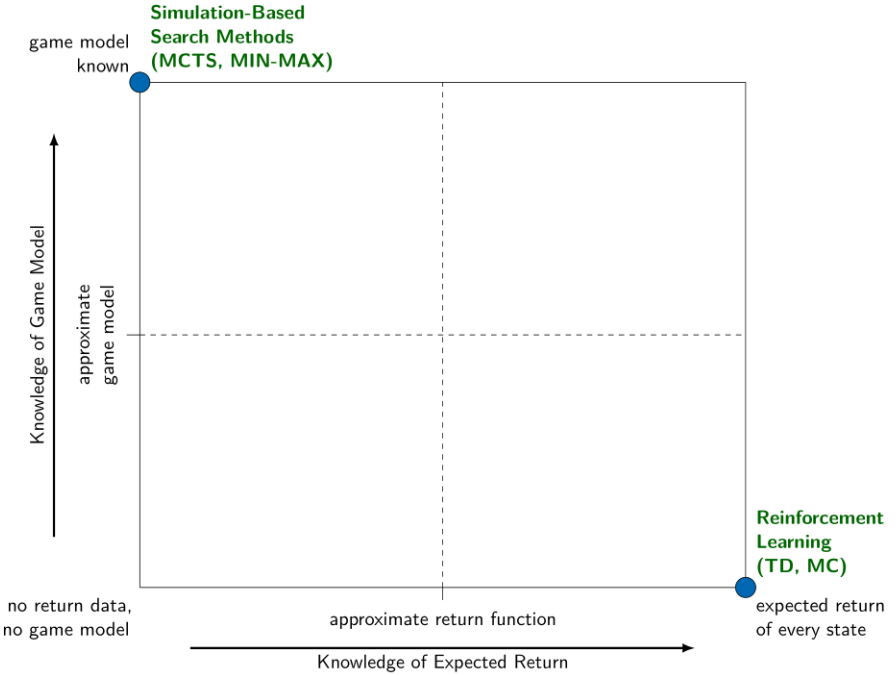
Most Popular Algorithm Classes

Reinforcement Learning

- Performance depends on the available training time

Simulation-based Search

- Performance depends on the available computation time during evaluation



OTTO VON GUERICKE UNIVERSITÄT MAGDEBURG

Queen Mary University of London

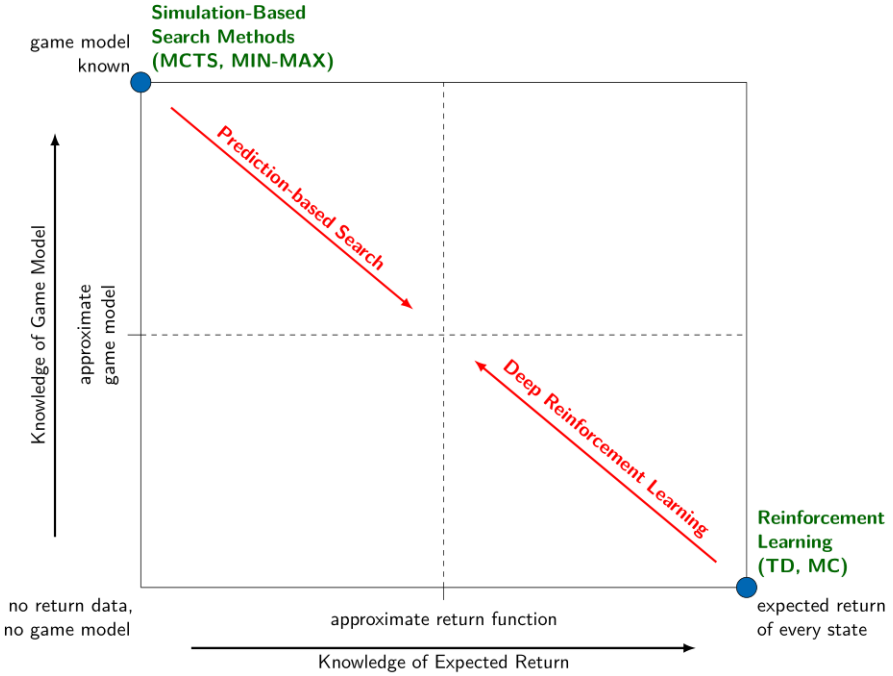
Current and Proposed Solution

Deep Reinforcement Learning

- Learn an approximate function that maps the state to the expected reward

Prediction-based Search

- Learn an approximate function that predicts the next state



OTTO VON GUERICKE UNIVERSITÄT MAGDEBURG

Queen Mary University of London

Forward Model Learning

Goal: Learn to predict the upcoming states of the environment.

A forward model fm maps the environment's state S_t and the agent's action A_t at time t to the upcoming state S_{t+1} of the environment

$$fm : (\mathcal{S} \times \mathcal{A}) \rightarrow \mathcal{S} \qquad (S_t, A_t) \longmapsto S_{t+1}$$

Above definition applies to environment models that fulfill the Markov Property:

$$P(S_{t+1} \mid S_0,\ A_0,\ S_1,\ A_1,\ \ldots,\ S_t,\ A_t) \quad \Rightarrow \quad P(S_{t+1} \mid S_t,\ A_t)$$

Model Requirements

Model accuracy:

- accurate predictions are required to simulate future time-steps

Model speed:

- the trained model needs to be fast to facilitate more simulations

Model size:

- the number of parameters should be low

Interpretability and Reliability:

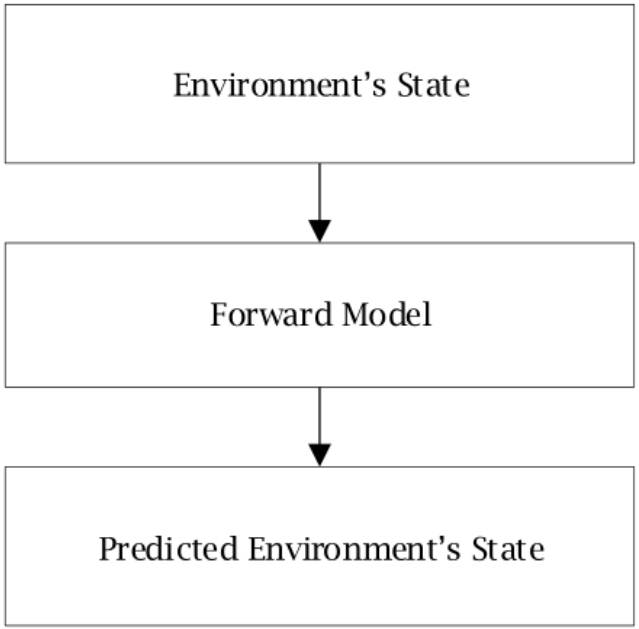
- risk aware applications require interpretable models

End–To–End Solution

Learning a model by consecutive interaction with the environment .

Each observed state transition represents a single training example.

- Classifiers and regressors can be used for the prediction depending on the type of output



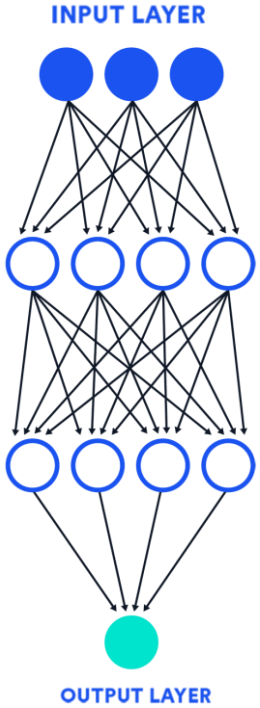
OTTO VON GUERICKE UNIVERSITÄT MAGDEBURG

Queen Mary University of London

End–To–End Solution

Deep Learning Forward Models

- Current systems often make use of deep neural networks to predict upcoming states
- Many training examples are required to fit the networks parameters



The training time is dependent on the size of the state and action space. Both can be enormous!

- How to reduce the model's training time?

OTTO VON GUERICKE UNIVERSITÄT MAGDEBURG

Queen Mary University of London

Decomposed Forward Model

Assumption:

- sensor values can be modelled independently

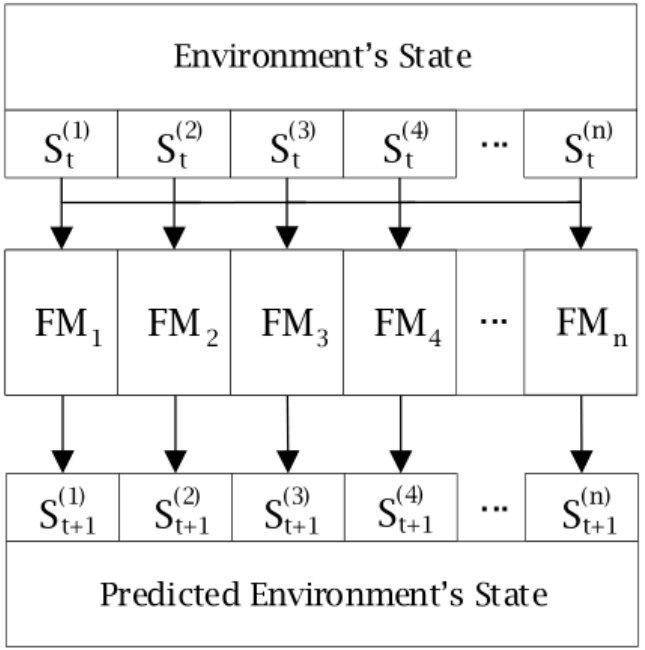
$$\forall i, j \in 1..n : i \neq j \Rightarrow S_{t+1}^{(i)} \perp\!\!\!\perp S_{t+1}^{(j)} \mid S_t, A_t$$

Learn on sub-model for each observable sensor value

$$\underset{i}{fm} : (S_t,\ A_t) \longmapsto S_{t+1}^{(i)}$$

Aggregate the result of each sensor value prediction

$$fm(S_t, A_t) = (fm_1(S_t, A_t),\ fm_2(S_t, A_t),\ \ldots,\ fm_n(S_t, A_t))$$
$$= (S_{t+1}^{(1)},\ S_{t+1}^{(2)},\ \ldots,\ S_{t+1}^{(n)}) = S_{t+1}$$



OTTO VON GUERICKE UNIVERSITÄT MAGDEBURG

Queen Mary University of London

Continuous State-Space Forward Models

The decomposed forward model can be used to model continuous state-spaces

- Instead of predicting the resulting state, predict the changes in between states

$$\text{i-th Component Model:} \quad p(s_t^i \mid s_{t-1}, a_{t-1}, \ldots, s_1, a_1)$$

$$\text{i-th Differential Model:} \quad p(s_{t-1}^i - s_t^i \mid s_{t-1}, a_{t-1}, \ldots, s_1, a_1)$$

Example: Predicting the movement of a robot

- *End-to-End*: Predict the robot's final position at time *t+1*
- *Differential Decomposed Model*: separately predict the change in position of its arms and legs

Optimal Decomposition

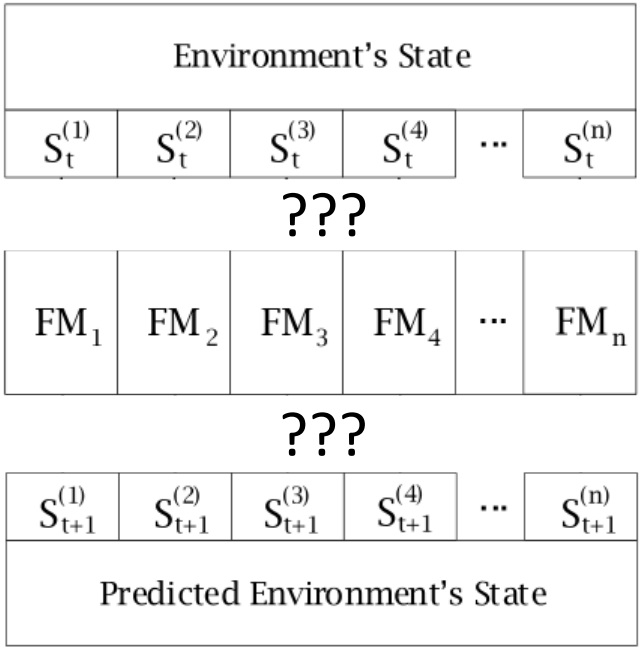
In case enough observation data is available:

- infer the dependencies among the in- and outputs sensor values
- optimal decomposition can be achieved

Alternatively, we can ask experts to model the variables dependencies or use heuristic solutions.

With each independency we:

- reduce the model space considerably
- require less training data
- create a more efficient model



OTTO VON GUERICKE UNIVERSITÄT MAGDEBURG

Queen Mary University of London

Model Decomposition Heuristic – Locality

Assumptions:

- structured representation of the state
- requires a similarity or distance function for sensor values
- semantic of a sensor-value is independent of its index

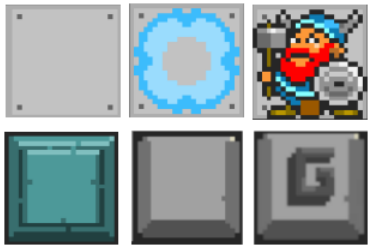
Game-State of Sokoban

Tile-based Representation (of Video Games):

- a state can be represented as a matrix T of size *n*

$$T = \begin{bmatrix} T(1,1) & \dots & T(1,m) \\ \vdots & \ddots & \vdots \\ T(n,1) & \dots & T(n,m) \end{bmatrix}$$

- $T(x, y)$ specifies the observed tile at position (x, y)



Tilemap Components

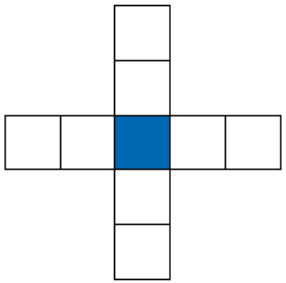
Local Decomposition

Decompose the forward model into one sub-model per tile:

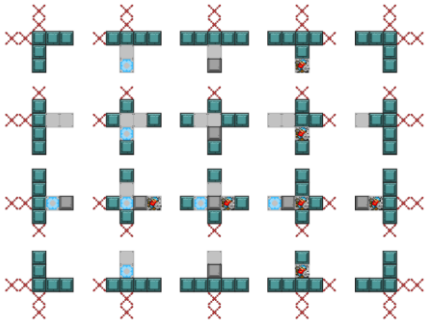
$$\underset{x,y}{fm} : \left(N(x,y)_t,\ A_t\right) \longmapsto T(x,y)_{t+1}$$

- $N(x,y)_t$ describes the local neighbourhood of tile $T(x,y)$ at time t
- it contains each tile with distance less than a given threshold

Game-State of Sokoban



Local Neighbourhood



Extracted Patterns

OTTO VON GUERICKE UNIVERSITÄT MAGDEBURG

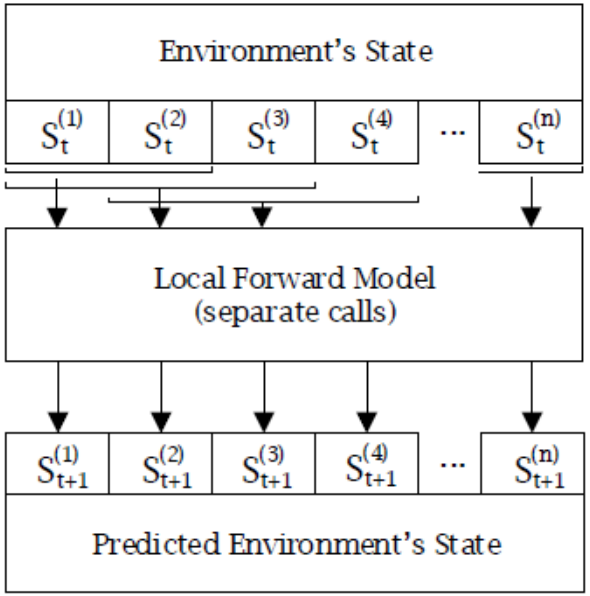
Queen Mary University of London

Local Forward Model

Predict the next state by predicting each tile

$$T_{t+1} = \begin{bmatrix} fm_{1,1}(N(1,1), A_t) & \dots & fm_{1,m}(N(1,m), A_t) \\ \vdots & \ddots & \vdots \\ fm_{n,1}(N(n,1), A_t) & \dots & fm_{n,m}(N(n,m), A_t) \end{bmatrix}$$

In case the semantic of a tile is independent of its position, only a single model needs to be learned



Advantage: higher sampling efficiency

- each observed state transition consists of one observed pattern per tile (in total: $n \times m$ patterns)

OTTO VON GUERICKE UNIVERSITÄT MAGDEBURG

Queen Mary University of London

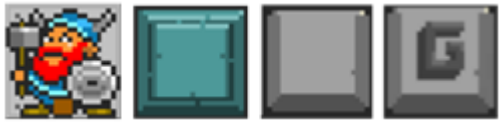
Model Decomposition Heuristic – Object Semantics

Object-based Representation:

- the state consists of multiple entities of which several attributes can be observed

$$\mathcal{S} = (S^{(1)}, S^{(2)}, \ldots, S^{(n)})$$
$$= (\underbrace{S^{(1,1)}, \ldots, S^{(1,i)}}_{\text{Object 1}}, \underbrace{S^{(2,1)}, \ldots, S^{(2,j)}}_{\text{Object 2}}, \ldots, \underbrace{S^{(m,1)}, \ldots, S^{(m,k)}}_{\text{Object m}})$$

- Changing Tiles can be understood as objects that change their position or appearance.



Tilemap Components that equal objects or characters that move

Object-based Decomposition

Assumptions:

- game components are considered to represent independently acting entities
- similar looking objects exhibit similar behavior

Create one model for each entity or entity type:

$$fm_i : \left((S_t^{(i,1)}, \ldots, S_t^{(i,k)}),\ A_t \right) \longmapsto (S_{t+1}^{(i,1)},\ \ldots,\ S_{t+1}^{(i,k)})$$

Complex entities can be modelled using a decomposed forward model

- create one model for each observable sensor value

$$fm_{i,j} : \left((S_t^{(i,1)}, \ldots, S_t^{(i,k)}),\ A_t \right) \longmapsto S_{t+1}^{(i,j)}$$

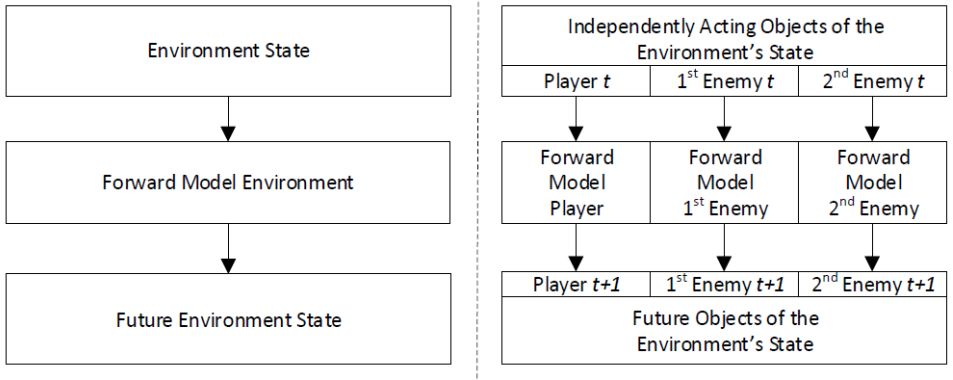
OTTO VON GUERICKE UNIVERSITÄT MAGDEBURG

Queen Mary University of London

Object-based Forward Model

Aggregate the prediction of each object and its associated sensor values:

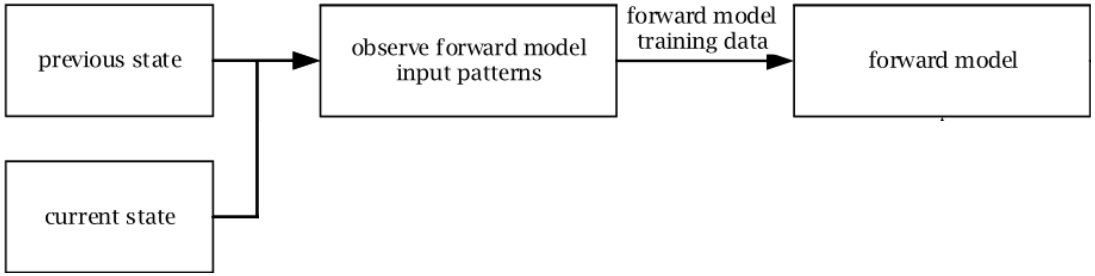
$$
\begin{aligned}
fm(S_t, A_t) &= (\underset{1}{fm}(S_t, A_t),\ \ldots,\ \underset{n}{fm}(S_t, A_t)) \\
&= ((\underset{1,1}{fm}((S_t^{(1,1)}, \ldots, S_t^{(1,k)}), A_t),\ \ldots,\ \underset{m,k}{fm}((S_t^{(m,1)}, \ldots, S_t^{(m,k')}), A_t))) \\
&= (S_{t+1}^{(1)},\ S_{t+1}^{(2)},\ \ldots,\ S_{t+1}^{(n)}) = S_{t+1}
\end{aligned}
$$



OTTO VON GUERICKE UNIVERSITÄT MAGDEBURG

Queen Mary University of London

Prediction-based Search



Comparison of Learning Approaches

(Deep) Reinforcement Learning:

- immediately output the action that is best according to previous experiences

Prediction-based Search:

- search for the best action-sequence according to the simulated environment

➢ Solution can be interpreted since the whole search tree is available

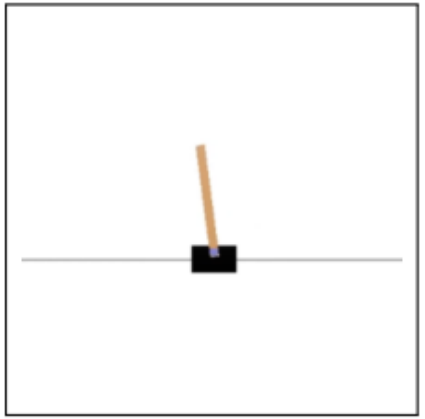
➢ Evaluating the model's prediction confidence and model the agent's risk

➢ Performance scales with the forward model's accuracy and the available search time

A Simple Example

A simple but traditional example: the Cart Pole problem

- **Task:** balance the pole by moving the cart left or right
- **Observation:** position and speed of the cart and the pole's tip
- **Actions:** moving left or right



Cart Pole Balancing Task

During training:

- explore unknown state and action pairs and learn a forward model

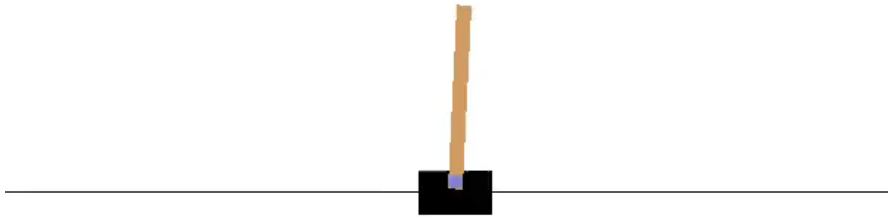
During evaluation:

- predict upcoming states and keep the pole balanced

OTTO VON GUERICKE UNIVERSITÄT MAGDEBURG

Queen Mary University of London

Video for Training and Evaluation Runs



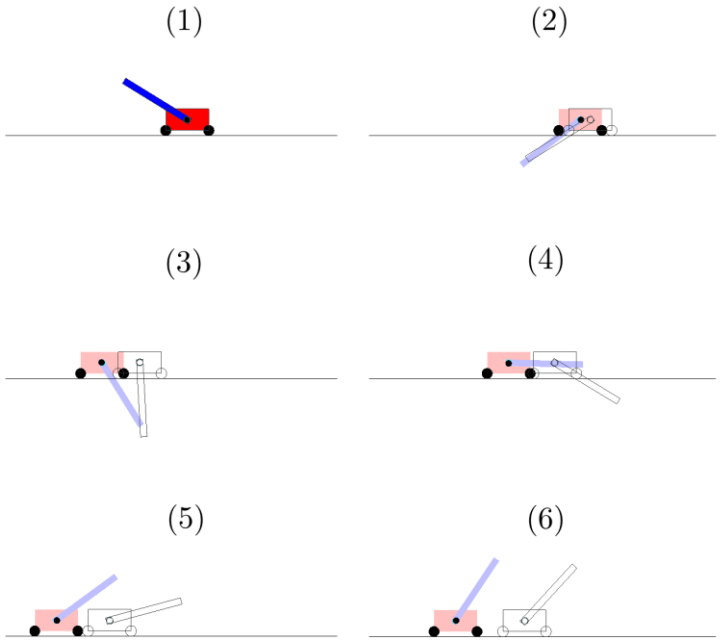
OTTO VON GUERICKE UNIVERSITÄT MAGDEBURG

Queen Mary University of London

Importance of the Prediction Accuracy

Comparing the true and the predicted state allows us to measure the models accuracy:

- Initial states are predicted well since many of them have been sampled during training
- Errors propagate over consecutive predictions
- Rare events or insufficiently sampled states can yield drastic prediction errors



Adapted from: Freeman, C. D., Metz, L., & Ha, D. (2019). Learning to Predict Without Looking Ahead: World Models Without Forward Prediction. 9, 1–12. http://arxiv.org/abs/1910.13038

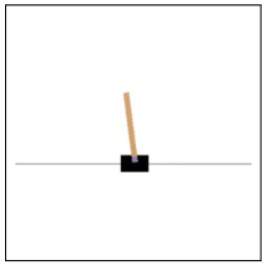
OTTO VON GUERICKE UNIVERSITÄT MAGDEBURG

Queen Mary University of London

Motion Control Test Environments

Five Motion Control Environments of the OpenAI Gym Framework gym.openai.com

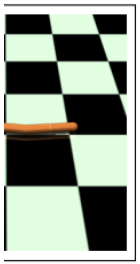
- Testing discrete in- and outputs as well as rewards
- All tested environments are fairly low in complexity
  - By studying them we want to achieve a clear picture on prediction-based search methods and how they perform



(a) Cart Pole

(b)

OpenAI Gym Environment	State Space	Action Space Dim	Action Space Type	Rewards
Cartpole-v1	$\mathbb{R}^4$	1	discrete	discrete
Acrobot-v1	$\mathbb{R}^6$	1	discrete	cont.
LunarLander-v2	$\mathbb{R}^8$	1	discrete	cont.
Pendulum-v0	$\mathbb{R}^3$	1	cont.	cont.
Swimmer-v2	$\mathbb{R}^6$	2	cont.	cont.



vimmer

OTTO VON GUERICKE UNIVERSITÄT MAGDEBURG

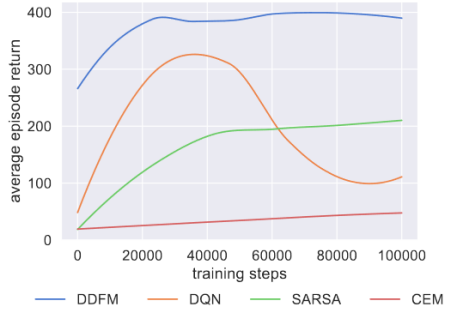
Queen Mary University of London

Training Results I/II

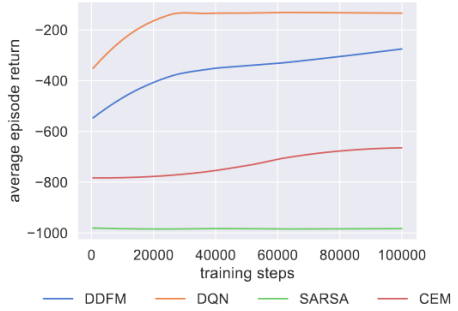
Both, **Cart Pole** and **Acrobot**, can be effectively learned and show stable results

First successful run in **Cart Pole** has been observed in the fourth episode.

In **Acrobot** the prediction accuracy slowly increases for longer search depths.



(a) Cart Pole



(b) Acrobot

OTTO VON GUERICKE UNIVERSITÄT MAGDEBURG

Queen Mary University of London

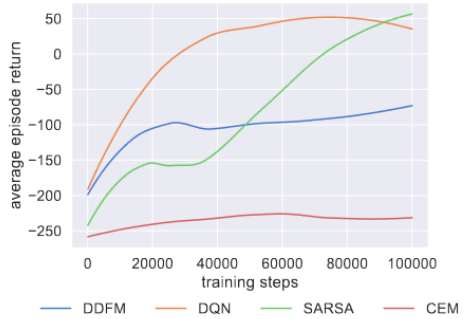
Training Results II/II

Lunar Lander: the agent quickly learn to land in case the target is directly below it

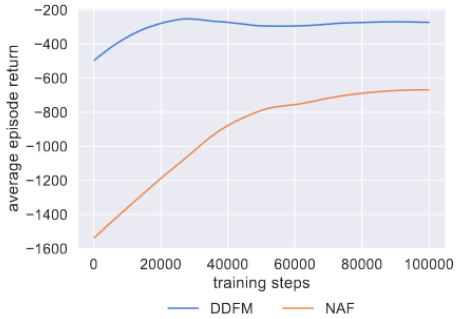
- But fails to land at distant positions due to missing positive examples

Pendulum: the agent quickly learns to swing up the pole and keeps it balanced

- With limited search depth, the agent does not swing to the opposite direction for speed



(c) Lunar Lander



(d) Pendulum

OTTO VON GUERICKE UNIVERSITÄT MAGDEBURG

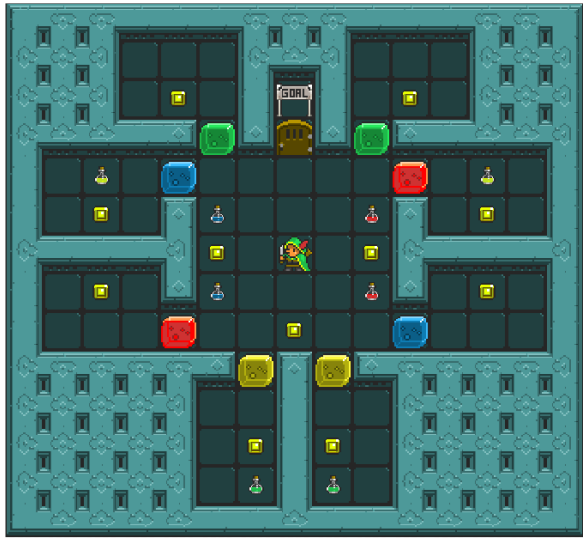
Queen Mary University of London

Evaluation Setup GVGAI Games I/II

The evaluation is based on 30 games of the General Video Game AI (GVGAI) framework

Varying Game Characteristics:

- types and number of NPCs
- use of a resource system
- reward style (dense/sparse)
- the number and types of termination conditions
- determinism vs. non-determinism
- the number of actions available



OTTO VON GUERICKE UNIVERSITÄT MAGDEBURG

Queen Mary University of London

Evaluation Setup GVGAI Games II/II

Local and Object-based Forward Models (LFM/OBFM) have been used in conjunction with:

- Breadth First Search (BFS)
- Rolling Horizon Evolutionary Algorithm (RHEA)
- Monte Carlo Tree Search (MCTS)

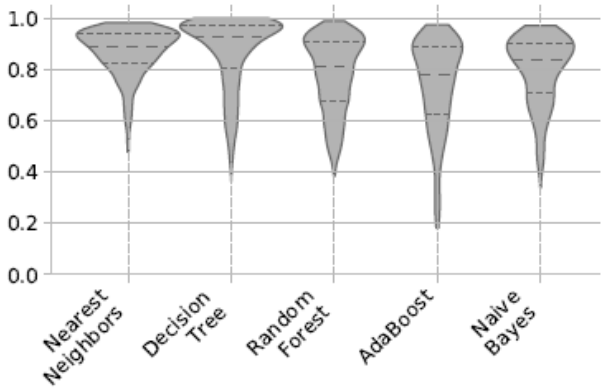
Trained agents are compared to a random agent

- in previous research competitions no agent performed significantly better
- given the agent has no prior experience and is unaware of the original forward model

Prediction Accuracy Over Time

Data Set Generation

- collection of observed state transitions of a random agent
- all 5 levels were played 10 times for 200 ticks each
- 9 different local neighborhood patterns were extracted (one data set each)



Accuracy Evaluation and Model Selection

- evaluation of 5 classifiers, multiple parameters

OTTO VON GUERICKE UNIVERSITÄT MAGDEBURG

Queen Mary University of London

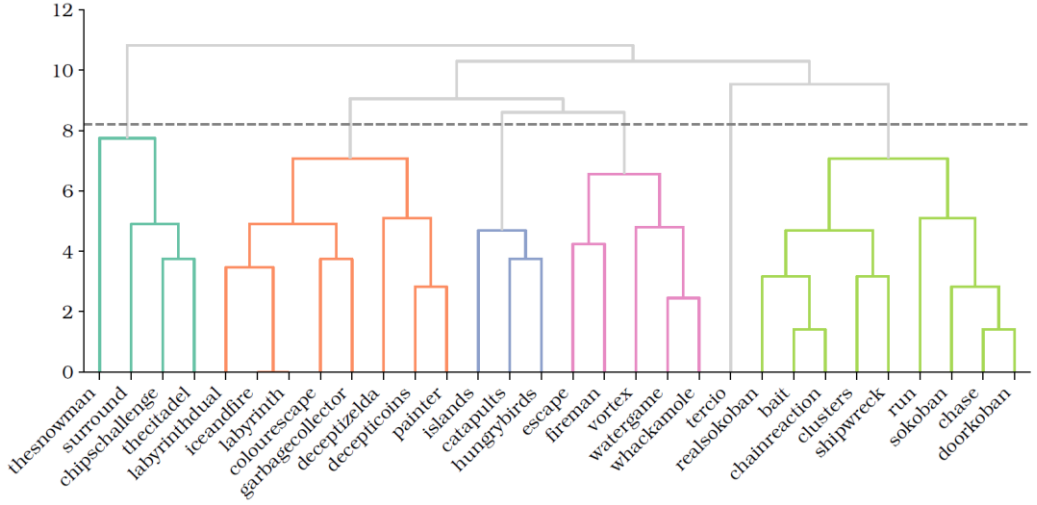
Prediction Accuracy Over Time

Agents are ranked according to their:

- average win-rate, average score, average ticks of won and lost games
- results are clustered to find groups of games in which the agent performs similar

Clusters represent:

- Sparse reward games
- Maze-like games
- Long-term planning games
- Games including randomness
- Puzzle games



OTTO VON GUERICKE UNIVERSITÄT MAGDEBURG

Queen Mary University of London

Prediction Accuracy Over Time

Aggregated ranks over all tested games and final score per agent

Agents		Rank 1st	2nd	3rd	4th	5th	6th	7th	Formula-1 Score
Random		4	0	1	3	3	4	15	303
LFM	BFS	10	5	4	4	2	2	3	502
	RHEA	3	3	3	8	3	10	0	380
	MCTS	5	3	8	4	2	4	4	423
OBFM	BFS	6	8	2	4	4	1	5	450
	RHEA	3	5	6	2	10	4	0	411
	MCTS	5	7	4	3	5	4	2	441

Formula-1 Scoring System: $1^{st} = 25$, $2^{nd} = 18$, $3^{rd} = 15$, $4th = 12$, $5th = 10$, $6th = 8$, $7th = 6$

OTTO VON GUERICKE UNIVERSITÄT MAGDEBURG

Queen Mary University of London

Planning Horizon Dilemma I/II

Fixing a planning horizon is a non-trivial task:

- Long planning horizons yield a more accurate estimation of the expected return
- but it also increases the number of states to be analyzed

In addition to this trade-off the model's accuracy limits the effective search depth

- As shown before, errors will accumulate over time reducing our confidence with every layer of the search tree.
- In comparison, the true forward model provides an accurate prediction for each level of the search tree.

Planning Horizon Dilemma II/II

Probabilistic classifiers may allow us to measure the model's confidence in made predictions

- in case we can measure the confidence of our prediction, we may bound the search to a confidence threshold
- therefore, dynamically limiting the search depth

Even better: learn models that are reliable in the long-term prediction task

Exploration vs. Exploitation during Training

The classic **exploration vs. exploitation** tradeoff affects the training procedure

- **exploration:** gathering samples of the whole state-space to assure an accurate prediction for all value ranges
- **exploitation:** focus on analyzing promising areas of the state-space to efficiently learn how to act in the unknown environment

Conclusion

Forward Model Learning can be feasible for motion control applications but lacks

- model centric training approaches
- long-term reliable forward models

Similarly, search-based methods can yield great performance but are missing

- methods for efficiently selecting actions in continuous action spaces

State- and action-abstraction methods may resolve these problems.

Future Work

- How does the presented approach compare with model-based reinforcement learning?
- Extend the efficiency of search-based algorithms in case of continuous state and action-spaces.
- Estimate the reliability of a model's prediction given the current observation.
- Implement application-centric state and action abstractions to speed-up the model building process.

Thank you for your attention!

Our Related Works

Dockhorn, A. (2020). Prediction-based Search for Autonomous Game-Playing. *PhD Thesis, Otto-von-Guericke University Magdeburg,* https://doi.org/10.25673/34014

Apeldoorn, D., & Dockhorn, A. (2020). Exception-Tolerant Hierarchical Knowledge Bases for Forward Model Learning. *IEEE Transactions on Games* (accepted). https://doi.org/10.1109/TG.2020.3008002

Dockhorn, A., Lucas, S. M., Volz, V., Bravi, I., Gaina, R. D., & Perez-Liebana, D. (2019). Learning Local Forward Models on Unforgiving Games. *2019 IEEE Conference on Games (CoG)*, 1–4. https://doi.org/10.1109/CIG.2019.8848044

Dockhorn, A., Tippelt, T., & Kruse, R. (2018). Model Decomposition for Forward Model Approximation. *2018 IEEE Symposium Series on Computational Intelligence (SSCI)*, 1751–1757. https://doi.org/10.1109/SSCI.2018.8628624

Rudolf Kruse
Faculty of Computer Science
Otto von Guericke University Magdeburg

Alexander Dockhorn
School of Electrical and Computer Engineering
Queen Mary University London